AD-A190 661

MEMORANDUM REPORT BRL-MR-3622

A TECHNIQUE FOR CODE AUGMENTATION

KURT D. FICKIE
JOHN GROSH

OCTOBER 1987

US ARMY BALLISTIC RESEARCH LABORATORY
ABERDEEN PROVING GROUND, MARYLAND

DESTRUCTION NOTICE

Destroy this report when it is no longer needed. DO NOT return it to the originator.

Additional copies of this report may be obtained from the National Technical Information Service, U.S. Department of Commerce, Springfield, VA 22161.

The findings of this report are not to be construed as an official Department of the Army position, unless so designated by other authorized documents.

The use of trade names or manufacturers' names in this report does not constitute indorsement of any commercial product.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188
Exp. Date: Jun 30, 1986

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
BRL-MR-3622

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
Ballistic Research Labortory

6b. OFFICE SYMBOL (If applicable)
SLCBR-IB-A

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS (City, State, and ZIP Code)
Aberdeen Proving Ground, MD 21005-5066

7b. ADDRESS (City, State, and ZIP Code)

8a. NAME OF FUNDING / SPONSORING ORGANIZATION

8b. OFFICE SYMBOL (If applicable)

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER

8c. ADDRESS (City, State, and ZIP Code)

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
61102A	1L161102AH43	00	00

11. TITLE (Include Security Classification)
A Technique for Code Augmentation

12. PERSONAL AUTHOR(S)
Kurt D. Fickie and John Grosh

13a. TYPE OF REPORT
Memorandum Report

13b. TIME COVERED
FROM Aug 87 TO Aug 87

14. DATE OF REPORT (Year, Month, Day)

15. PAGE COUNT

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
19	01	
21	02	

18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
Optimization, Computer Aided Design, IBHVG2, BLAKE

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

A simple method for calling pre-existing computer codes from inside another program is described. Three applications drawn from the field of interior ballistics are included as examples. Two of the cases are optimization problems and the other is a simple search for a constraint condition. More elaborate applications in the area of computer aided design (CAD) are discussed.

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
John Grosh

22b. TELEPHONE (Include Area Code)
(301) 278-6100

22c. OFFICE SYMBOL
SLCBR-IB-P

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

CONTENTS

List of Figures

I. INTRODUCTION

Researchers involved in product development regularly desire to modify or extend computer models. Usually these changes or applications were unforeseen when the code was originally developed. Altering the code to accommodate the researcher represents the obvious and traditional solution. For a large, complex code this represents a formidable, and frequently frustrating, task. The situation is exacerbated if the code authors are unavailable, the algorithm is poorly documented, the code is unstructured, etc. Furthermore, some codes are extremely fragile due to obsolete programming practices. Seemingly small or insignificant changes often will break the code.

Proprietary codes represent another difficulty. In some situations source code is specifically *not* provided for a variety of reasons. This scenario usually dictates a negotiated contract to implement new changes or features. Besides being untimely, the process leaves little margin for error or dampens the desire to experiment with new ideas.

Faced with these difficulties, the program is rewritten more often than not. In the meantime, some calculations are avoided altogether or are performed via tedious cycles. There are alternatives to this expensive proposition. It is quite possible to utilize an established computer model as an isolated black box and write another code which merely calls it as a programming sub-unit. The method has many advantages. Not directly tampering with the innards of the black box is just one. Another strength is that it tends to modularize the problem. Changes are so easy to understand and implement that it encourages further investigation and algorithm refinement.

This report will illustrate this technique using the popular interior ballistics codes IBHVG2 [1] and BLAKE [2]. Some problems will be posed and solved. These case studies were designed to be instructive, so they were deliberately simple constructions. Still, we feel they represent nontrivial extensions to the respective codes and are typical of calculations which are currently performed by hand. The methodology is quite general, however. The reader is encouraged to expand on these examples and think up new applications.

II. IMPLEMENTATION

Calling one compiled code (already executable) from another can usually be accomplished via system calls. Since the Department of Defense has largely converted to the UNIX operating system, our example will explicitly cover this circumstance. Other operating systems will typically allow the same approach but the details will vary.

Figure 1 diagrams the process. The main program drives the problem. In our simple examples, it declares storage, initializes variables, and calls a subprogram, SUB1, to perform the mathematics. This program makes a system call which runs MAIN2, our "black box" program, as a subordinate unit. The results of this run are available to SUB1. We hope the complete case examples which follow will clarify the technique.

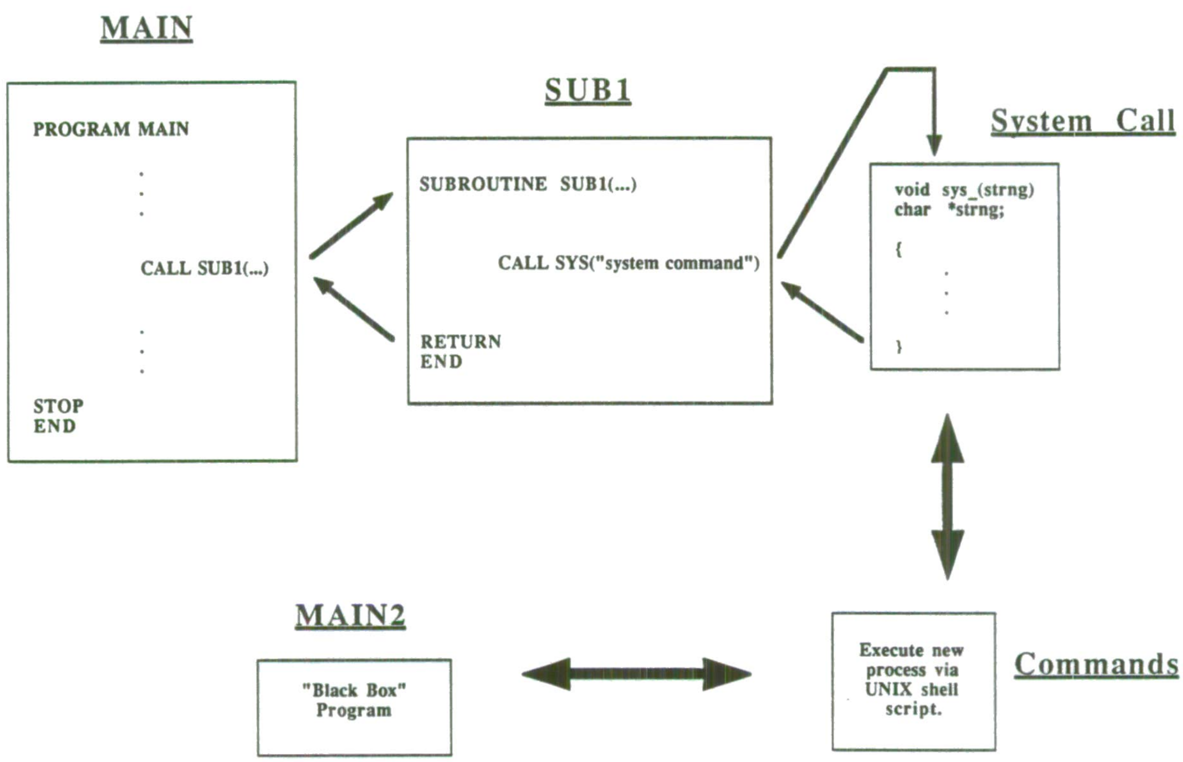

Figure 1. "Black Box" Flowsheet

A. An Optimization Example

A charge designer frequently must maximize the projectile velocity given a set of constraints. Usually, a number of variables are available which may be manipulated to increase the velocity. Propellant mix, grain dimensions, and charge weight are typical examples. Velocities which result in a violation of a design constraint are infeasible. As an illustration, a set of inputs resulting in a breech pressure exceeding safety specifications will be excluded. This application is a classic constrained optimization problem.

The example problem involves optimizing the performance of a 120-mm gun system using a 19-perf, granular propelling charge. For this system, propellant characteristics are varied to maximize the velocity. The grain is characterized by setting the length and grain diameter (assuming perforation diameter to be fixed). The remaining free variable is the total charge weight. All other system parameters remain fixed. The constraints are easy to understand. Clearly, the designer cannot specify more propellant than can actually fit in the gun chamber. How much you can load will be very dependent on grain dimensions, however. In other words, for a set of inputs to be feasible, the charge weight must not exceed the maximum loadable weight for a given grain type. Furthermore, the resulting maximum breech pressure must fall below the given design specification.

Table 1. Notation for Optimization Problem

Symbol	*IBHVG2 Variable*	*Description*
l	LEN	grain length
D	DIAM	grain diameter
C	CHWT	propellant charge weight
C^*	-	maximum loadable charge weight
V_c	CHAM	chamber volume available for propelling charge
P^*	-	maximum breech pressure for a given set of inputs
P_M	PMAX	maximum allowable breech pressure (design variable)
v	VMUZ	muzzle velocity of projectile

Using the notation of the previous table, this problem can be simply stated mathematically as

$$\text{Maximize:} \qquad v = g(l,D) \tag{A.1}$$

subject to the constraints

$$P_M \geq P^* \left[l,D,C \mid C \leq C^*(l,D) \right] \tag{A.2}$$

Some simple sketches should clarify the constraints. Maximum breech pressure is assumed to be a monotonically increasing function of charge weight for a given grain type. In this case, there will be two possibilities. Figure 2 illustrates the situation for a given grain size (l,D) where the maximum loadable charge weight, C^*, results in a maximum pressure below the design limitation. Figure 3 depicts the case where the pressure limit dictates a loading below C^*. If one further assumes that muzzle velocity is a monotonically increasing function of maximum pressure for a given grain type, then the optimization domain would not include C since it is not a free variable. The optimal C would clearly be $C^*(l,D)$ for the first case and $f^{-1}(P_M)$ for the second case given that $P^* \equiv f(C)$.

Table 2. Program Organization

Figure 1	*Example 1*	*Description*
MAIN	OPTIMZ	main program for optimization
SUB1	FUNC	user-written routine called by ZXMIN (IMSL routine)
sys_	SYS	C-code to perform system call
Commands	run.ibhvg2	operating system commands to execute "black box"
MAIN2	IBHVG2	"black box" code

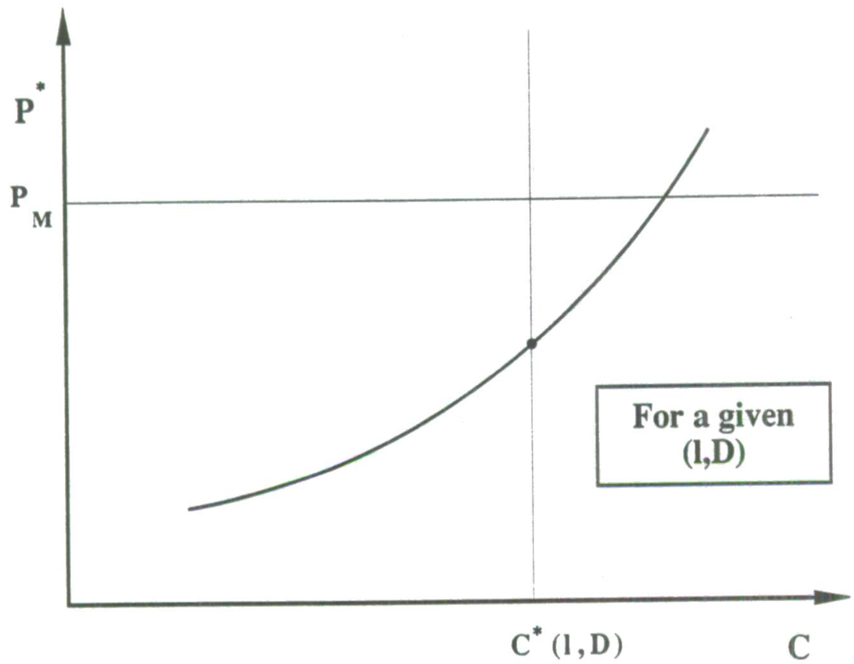

Figure 2. Maximum Loading Limitation

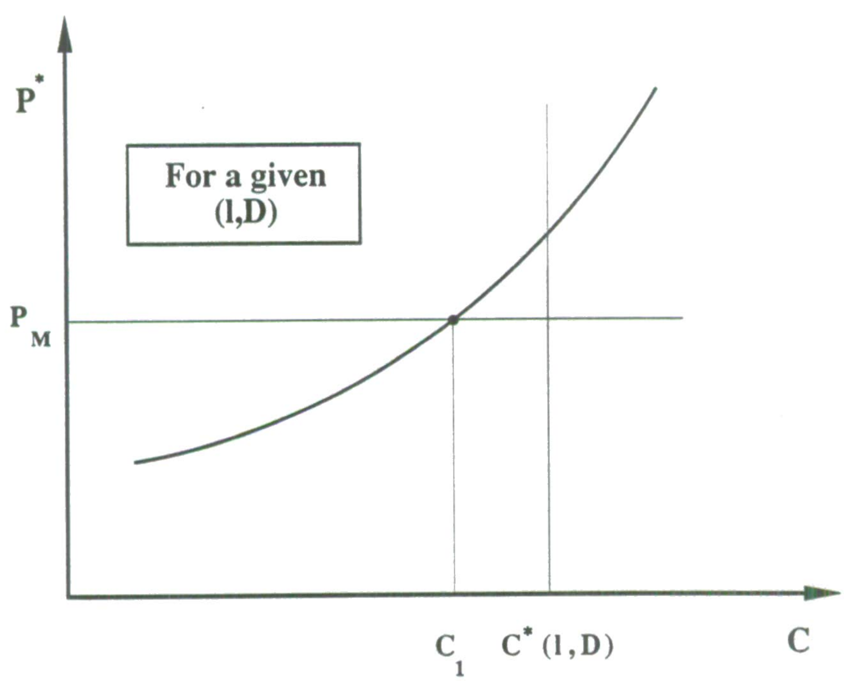

Figure 3. Design Pressure Limitation

Now we will discuss the actual coding. In our specific case, we desire to optimize projectile velocity subject to the constraint of 79 Kpsi for a maximum pressure. The manipulative variables will be grain diameter and length. All other inputs which define the interior ballistics remain constant. This problem was solved using a small, user-written program, a UNIX shell script, and the International Mathematical Subroutine Library (IMSL) [3]. We will discuss each in detail.

The main program, OPTIMZ, primarily does one thing: it calls the optimization routine, ZXMIN, from IMSL. It is quite short because ZXMIN performs the real work. This was intentional. IMSL is a commercial package popular to many scientific organizations. The routines are supported by a professional staff and are extensively tested. We chose IMSL because it is nearly ubiquitous; however, there are many other high quality libraries to pick from. Some recommended alternatives are: NAG, MINOS, MINPACK, LINPACK, CMLIB, TOMS, Harwell, PORT, and SLATEC (see [4] for more information). Whatever the choice, we feel that if at all possible, the engineer should avail himself of an immense amount of work by merely linking to pre-tested routines. Chosing the right algorithm for the job can be challanging in itself; conducting original research in optimization theory is usually counter-productive and better left in the hands of an expert. Furthermore, the documentation and verification process is simplified because one may reference the corresponding library.

OPTIMZ

```
      program optimz

      external func
      integer ier, n, nsig, iopt, maxfn
      real g(2), h(13), v, x(2), w(16)
c
c        Initialize
c
      n = 2                # dimension of domain
      nsig = 3             # number of significant digits
      maxfn = 100          # max number of function evaluations
      iopt = 3             # options selector (See IMSL manual)
      x(1) = 0.357         # grain diameter --  initial guess
      x(2) = 0.408         # slot size      --  initial guess
c
c        IMSL routine for function minimization
c
      call zxmin(func, n, nsig, maxfn, iopt, x, h, g, v, w, ier)

      write (6,'(f10.5,f10.5,f10.5)') x(1), x(2), v
      write (6,'(a,i3)') 'ier = ',ier
      end
```

ZXMIN uses a quasi-Newton method to find the minimum of scalar function of N variables. We found that computing the Hessian matrix produces a conservative but controlled optimization (IOPT=3) for this type of problem. It should be added that there are numerous optimiza-

tion algorithms available. We used ZXMIN merely for convenience. No effort was made to customize an algorithm for the type of topology found in ballistic problems. Some useful alternative candidates might be the *downhill simplex method* or one of the *direction set methods* [5,6,7]. These should execute much faster because no derivatives are estimated numerically.

ZXMIN requires a user-supplied subroutine to compute the scalar function to be minimized. FUNC shown below performs our particular task. Since we desire to maximize the velocity, we returned the negative value to ZXMIN. FUNC is rather unorthodox, but forms the crux of this report. FUNC serves as the bridge to the "black box" code, IBHVG2. The role of the subroutine is to generate a string from the arguments passed into it, and with that string, execute a UNIX shell script. Our Fortran compiler does not allow such a system call, so we link in a C program called SYS which will.

FUNC

```
      SUBROUTINE FUNC(N, X, V)
c
c   call a UNIX shell script which executes the IBHVG2
c   computer code with two parameters:
c
c       X(1) -- grain length
c       X(2) -- grain diameter
c
c   and returns the value V (muzzle velocity)
c

      INTEGER N
      REAL    X(N),V,P
      CHARACTER*120 CMD
      CHARACTER*25 STORE1
      CHARACTER*25 STORE2

c
c  using internal writes to convert the floating point
c  numbers to a character string
c
      WRITE( STORE1, '(f25.9)') X(1)
      WRITE( STORE2, '(f25.9)') X(2)
      CMD='run.ibhvg2 '//STORE1//'  '//STORE2

c
c          UNIX system call
c
      CALL SYS(CMD)

      OPEN(9,file='ibhvg2.out')
      REWIND(9)
      READ(9,*) P
      READ(9,*) V
      READ(9,*) C

      write(*,'(f10.7,f10.7,f10.1,f10.1,f10.3)') X(1),X(2),V,P,C

      V=-V

      RETURN
      END
```

SYS

```
/*  this C function is used to call a UNIX shell command (arbitrarily
    set to 120 characters) and is included because F77 does not have
    a similar facility
*/

void SYS(strng)
char          *strng;
{
              *(strng+119)=' ';  /* force the string to be null terminated */

              system(strng);
}
```

The UNIX shell script, *run.ibhvg2*, executes the IBHVG2 code. Prior to this, it modifies an input deck called *ibhvg2.in* to account for new input arguments (l and D). Muzzle velocity, maximum pressure, and charge weight are passed back into the main program via the output file *ibhvg2.out*. This is rather awkward, but the method is very general; it should work with most any operating systems. UNIX provides a more elegant approach through the use of "pipes" which are described briefly in the appendix.

run.ibhvg2

```
#    ===============================================================================================
#    Purpose:  Filter for IBHVG2 Input and Output:
#    Filename: run.ibhvg2
#    Date:     22 July 1987
#    ===============================================================================================
#
#    GENERAL SCRIPT INFORMATION:
#    ---------------------------
#    Shell script input arguments:   $1 - DIAM     ( grain diameter in main propellant deck)
#                                    $2 - LEN      ( grain length in main propellant deck)
#
#    IBHVG2 input filename: ibhvg2.in
#
#    Shell script output file: ibhvg2.out
#      - ibhvg2.out contains the values for maximum pressure (PMAX), muzzle velocity (VMUZ),
#        and the charge weight of the main propelling charge (CHWT), respectively, on
#        separate lines.
#
#    ALGORITHM:
#    ----------
#
#    This UNIX shell script will insert the values for LEN and DIAM and calculate and insert the
#    values for CHWT and web (WEB) into the IBHVG2 input file.  This input file is directed
#    into the IBHVG2 program.  The values for VMUZ, PMAX, and CHWT are filtered from output
#    of IBHVG2 and placed into this script's output file.
#
#    Charge weight will be  determined as a function of its diameter, length, and, in some
#    cases, as a function of the maximum allowable  chamber pressure, using the following algorithm:
#
#    1)  IBHVG2 will adjust CHWT so that a PMAX of 79000 psi is obtained (by using
#        the PMAX deck option in IBHVG2).
#    2)  The maximum loadable charge weight (CHWT_MAX) is calculated as a function of LEN
#        and DIAM (and system constants) from an empirical formula provided by
#        Kevin Resnik.
#    3)  If the CHWT obtained from Step 1. is less than or or equal to CHWT_MAX,
#        the results obtained from the IBHVG2 run in Step 1. are used as the output.
#
#        If the CHWT obtained from Step 1. is greater than the  CHWT_MAX, IBHVG2 will
#        be rerun using CHWT_MAX as the actual charge weight for the given set of inputs.
```

```
#
#   NOTES OF CAUTION:
#   -----------------
#   This shell script is NOT bullet-proof.  The IBHVG2 input file must be
#   set up with the following restrictions:
#     a) The main propellant deck must be placed in the second $PROP deck.
#     b) Each parameter varied by this shell script may SOLELY occupy
#        one line of the IBHVG2 input file.
#     c) For each parameter varied by this shell script, there must be
#        only one space before and after the equals sign.
#
#   ==========================================================================================
#
#   Section 1:  Run IBHVG2 for for CHWT which results from a PMAX deck.
#
#   ------------------------------------------------------------------------------------------


  cp ibhvg2.in  input1

# Split input file into sections so that decks can be easily altered using sed command.

  csplit -k -s -f part  input1  '/\$PROP/' {9}  2>/dev/null

  mv part00 header.deck
  mv part01 prop1.deck
  mv part02 tmp2.deck
  cat part0[3-9] > tail.deck

# Calculate web size in order to input parameter into IBHVG2 input deck. ( PD = perf diameter )

  PD=0.015
  WEB=`echo "($1 - 5.0*0.015)/6.0" | bc -l`

# Substitute grain parameters into main propellant deck.

  cat tmp2.deck | sed  's/DIAM = .*/DIAM = '$1'/p' |
                  sed  's/LEN = .*/LEN = '$2'/p'|
                  sed  's/WEB = .*/WEB = '$WEB'/p' > prop2.deck

# Combine input decks into one IBHVG2 input file.

  cat header.deck prop1.deck prop2.deck tail.deck > input1.deck

# Redirect input file into IBHVG2.

  IBHVG2 < input1.deck > output  2>/dev/null

# Filter out pertinent data from output file.

  PMAX=`grep "BREECH PRESS" output | awk '{ printf "%10.3f", $4 }'`
  VMUZ=`grep "VELOCITY" output  | awk '{ printf "%30.20f", $4 }'`
  CHWT=`sed -n '/- CHARGE 2 -/,/- CHARGE 3 -/p' output | grep WEIGHT | awk '{ print $8 }'`

#   ------------------------------------------------------------------------------------------
#   Section 2:  Determination of maximum loadable charge weight using Kevin Resnik's
#               empirical charge loading formula for 19 - perf cylindrical propellant.
#
#   ------------------------------------------------------------------------------------------

   DIAM=$1; LEN=$2; PD=0.015; VC=544.0; RHO=0.06

   PERCENT=`echo -13.125LEN + 68.125 | bc -l`
   RHOLOAD=`echo "(($RHO)*($DIAM*$DIAM - 19.0*$PD*$PD))/($DIAM*$DIAM)" | bc -l`
   CHWT_MAX=`echo "$PERCENT*$RHOLOAD*$VC/100.0 " | bc -l`
```

```
# -----------------------------------------------------------------------------------------
#
#   Section 3: If CHWT from PMAX deck > CHWT_MAX, then  IBHVG2 is rerun  using
#              CHWT_MAX as the actual CHWT.
#
#
# -----------------------------------------------------------------------------------------

#   Compare CHWT to CHWT_MAX.

if [ `echo "$CHWT"` -gt `echo "$CHWT_MAX"` ]
then

   # Prepare new IBHVG2 input deck with CHWT_MAX as the actual charge weight used.

   sed 's/CHWT = .*/CHWT = '$CHWT_MAX'/p' prop2.deck > prop2a.deck

   # Remove $PMAX option from IBHVG2 input deck.

   sed 's/\$PMAX/\$COMM/'  tail.deck                > taila.deck

   # Combine input decks into an IBHVG2 input file.

   cat header.deck prop1.deck prop2a.deck taila.deck > input2.deck

   # Redirect input file  into IBHVG2 program.

   IBHVG2  < input2.deck > output 2>/dev/null

   #  Filter out pertinent data from IBHVG2 output file.

   PMAX=`grep "BREECH PRESS" output | awk '{ printf "%10.3f", $4 }'`
   VMUZ=`grep "VELOCITY" output     | awk '{ printf "%30.20f", $4 }'`
   CHWT=`sed -n '/- CHARGE 2 -/,/- CHARGE 3 -/p' output | grep WEIGHT | awk '{ print $8 }'`

fi

#---------------------------------------- Remove Temporary Files ---------------------------------

rm *.deck output* part*  input*

#-------------------------------------------- FINAL OUTPUT -----------------------------------------
# write output data to ibhvg2.out

echo $PMAX   >ibhvg2.out
echo $VMUZ  >>ibhvg2.out
echo $CHWT  >>ibhvg2.out

#------------------------------------------------------------------------------------------------
```

The script modifies the input file shown on the following page and executes IBHVG2 with these incorporated changes. It should also be noted that the feasibility of the charge weight is examined inside the script. If the charge weight corresponding to the maximum design pressure exceeds what is physically possible to include within the chamber (see Figure 2), then the maximum loading weight will be used instead.

ibhvg2.in

```
$COMM
      INPUT FILE FOR OPTIMIZATION TEST CASE
$HEAT
      TSHL = 0.00450        CSHL = 1848       RSHL = 0.284
      TWAL = 293            HO   = 0.0648     HL   = 1
$GUN
      NAME = 'GERMAN 120MM GUN'      CHAM = 584          GRVE = 4.724
      LAND = 4.724     G/L   = 1.    TRAV = 187.11       TWST = 99
$PROJ
      NAME = 'SLUG'       PRWT = 18.00
$RESI
      NPTS = 4              AIR  = 1
      TRAV =        0,       .8,       3.0,     187
      PRES =     100,    2500,       100,     100
$INFO
      RUN  = 'OPTIMIZE'    DELT = 5E-5          DELP = 5E-5        EPS  = 0.002
      GRAD = 2             POPT = 1,0,1,0,0     SOPT = 0           CONP = 0
$RECO
      NAME = 'NONE'        RECO = 0             RCWT = 0

$PRIM        $ Primer Action  -  4.96 percent of total benite primer charge

      NAME = 'BENITE'      CHWT = 0.00347       GAMA = 1.25
      FORC = 212500        COV  = 30            TEMP = 2000

$PROP        $ Ignitor Action - 95.04 percent of total benite primer charge

      NAME = 'BENITE'      CHWT = 0.06653       GRAN = 'CORD'      IGNC = 0
      RHO  = 0.06          GAMA = 1.25          FORC = 212500      EROS = 0.000
      COV  = 30            TEMP = 2000          ALPH = 0           BETA = 27
      LEN  = 9.998         DIAM = 0.078

$PROP        $  Main Propelling Charge

      NAME = 'MAIN'        GRAN = '19P'         GAMA = 1.257       FORC = 395000
      COV  = 31.22         TEMP = 3049          EROS = 0.0000      RHO  = 0.06
      PD = 0.015           NTBL=5
      PR4L=   2000,    4000,    10000,    15000,    25000
      BR4L=  0.292,   0.856,    1.904,    2.143,    3.394
      CHWT = 19.10           $
      LEN = 0.408            $  Remember: One and only one space before and
      DIAM = 0.357           $            and after the equals sign for the
      WEB = 0.047            $            variables CHWT,LEN,DIAM, & WEB.

$PROP       $  Combustible Case - Modeled as a deterred propellant.

      NAME = 'FNC CASE'      CHWT = 1.41      GRAN = '1PF'    RHO  = 0.04
      COV  = 27.927          TEMP = 1610      EROS = 0.00     GAMA = 1.258
      FRCP= ,150000,150000   FRCE= ,150000,150000 FRCL(4)=200000
      IGNS(3)=2              THRS(3)=200      DEPE= , .015 , .0155
      NTBL=2                 DEPP= , .015 , .0155
      PR2P=1000,10000        BR2P=.5,2.4      PR3P=1000,10000  BR3P=.5,2.4
      PR2E=1000,10000        BR2E=.5,2.4      PR3E=1000,10000  BR3E=.5,2.4
      PR4L=1000,10000        BR4L=.5,10
      LEN  =  18             DIAM = 6.17      PD   = 6.01     WI =    .08

$PROP       $  Combustible Adapter

      NAME = 'KRAFT CASE'  CHWT = .21     GRAN = '1PF'    RHO  = 0.04     GAMA = 1.2734     FORC = 95726
      COV  = 9.883         TEMP = 1054    EROS = 0.00     ALPH =  1       BETA = 0.00001    IGNC = 0
      LEN  = 3.4           DIAM = 6.17    PD   = 6.01     WI =    .08

$PMAX
      VARY = 'CHWT'        NTH  = 2       TRY1 = 18.0     TRY2 = 19.0
      PMAX = 79000         EPS  = 5       LOOP = 100
$END
```

Table 3. Optimization Results

Iteration	Grain Diameter *(inches)*	Grain Length *(inches)*	Muzzle Velocity *(ft/s)*	Maximum Pressure *(psi)*	Charge Weight *(lbs)*
1	0.3570000	0.4080000	5344.6	78996.0	19.891
2	0.3681563	0.4080000	5138.8	71468.0	19.842
3	0.3458437	0.4080000	5368.1	79000.0	19.423
4	0.3570000	0.4207501	5347.4	78999.0	19.930
5	0.3570000	0.3952499	5340.1	79000.0	19.850
6	0.3681563	0.4207501	5107.5	70137.0	19.789
7	0.3681563	0.3952499	5171.5	72873.0	19.895
8	0.3458437	0.3952499	5363.8	79000.0	19.385
9	0.3458437	0.4207501	5368.2	79000.0	19.460
10	0.3573486	0.4080000	5342.3	78998.0	19.905
11	0.3570000	0.4083984	5344.8	78996.0	19.892
12	0.3370761	0.5953711	5263.1	71942.0	18.945
13	0.3470381	0.5016855	5388.9	79000.0	19.702
14	0.3473770	0.5016855	5387.7	79000.0	19.717
15	0.3470381	0.5021755	5389.0	79000.0	19.703
16	0.3349719	0.6175594	5256.1	71512.0	18.844
17	0.3410050	0.5596225	5406.1	78996.0	19.529
18	0.3413380	0.5596225	5404.1	78995.0	19.545
19	0.3410050	0.5601690	5406.2	78996.0	19.530
20	0.3361544	0.6030530	5261.0	71913.0	18.909
21	0.3385797	0.5813377	5411.8	79003.0	19.447
22	0.3389104	0.5813377	5412.8	78999.0	19.464
23	0.3385797	0.5819055	5411.9	79004.0	19.448
24	0.3390746	0.5856780	5248.6	71472.0	18.993
25	0.3388272	0.5835079	5412.8	78999.0	19.463
26	0.3391580	0.5835079	5412.0	79000.0	19.479
27	0.3388272	0.5840777	5254.7	71753.0	18.999
28	0.3388284	0.5835007	5412.8	78999.0	19.463
29	0.3388296	0.5834935	5412.8	78999.0	19.463
30	0.3388320	0.5834790	5412.8	78999.0	19.463
31	0.3388368	0.5834502	5412.9	79000.0	19.463
32	0.3388464	0.5833925	5412.9	79000.0	19.464
33	0.3391773	0.5833925	5412.0	78998.0	19.479
34	0.3388464	0.5839623	5254.6	71751.0	19.000
35	0.3388465	0.5833916	5412.9	79000.0	19.464
36	0.3388466	0.5833908	5412.9	79000.0	19.464
37	0.3391774	0.5833916	5412.0	78997.0	19.479
38	0.3388465	0.5839614	5254.6	71751.0	19.000
39	0.3388465	0.5833916	5412.9	79000.0	19.464
40	0.3391774	0.5833916	5412.0	78997.0	19.479
41	0.3385156	0.5833916	5411.6	79001.0	19.441
42	0.3388465	0.5839614	5254.6	71751.0	19.000
43	0.3388465	0.5828219	5412.8	79000.0	19.463
44	0.3389031	0.5832820	5413.0	78997.0	19.466
45	0.3389598	0.5831724	5413.2	78998.0	19.469
46	0.3390731	0.5829533	5411.8	78999.0	19.474
47	0.3392908	0.5831724	5412.1	78995.0	19.484
48	0.3386288	0.5831724	5412.1	79003.0	19.452
49	0.3389598	0.5837420	5413.3	78998.0	19.470

The results of the optimization are shown in Table 3. As can be seen, the program finds the crest region and then searches slowly due to the flat slope near the top. Clearly, this would be very tedious to perform by hand. Increasing the number of variables to be optimized would quickly render the problem intractable without a computational scheme *and* a fast computer. To provide some feel for the computational effort, this run requires about five minutes on a supercomputer and about two hours on a minicomputer.

B. Two Thermodynamic Examples

The BLAKE code computes thermodynamic and equilibrium information for hot propelling gases in a ballistic environment. There are many circumstances where running the BLAKE code in an iterative fashion would be useful. Our first case modifies the concentration of a two-component propellant to search for the combination of maximum total impetus.

Table 4. Program Organization for Case 1

Figure 1	*Example 1*	*Description*
MAIN	MAXIMP	main program for optimization
SUB1	F1	user-written routine called by ZXLSF (IMSL routine)
SYS	sys	C-code to perform system call
Commands	run.blake	operating system commands to execute "black box"
MAIN2	BLAKE	"black box" code

The idea is virtually the same as before. The main program sets up the problem, an IMSL routine does the work which executes a shell script. The IMSL routine for this problem, ZXLSF, performs a one-dimensional minimization of a smooth function using a safeguarded quadratic interpolation. The results are shown graphically in Figure 4. The coding follows.

MAXIMP

```
      program maximp

      external f
      integer ier, maxfn
      real f, xacc, x, step, bound

      x     =  50.0     # first guess, 10.0=50+step
      step  = -40.0
      bound =  49.0     # 1.0 <= x <= 99.0
      xacc  =   0.01    # accuracy
      maxfn =  50       # maximum function evaluations

      call zxlsf(f1, x, step, bound, xacc, maxfn, ier)
      write (6, '(a,f11.5,a,i3)' )'X=',x,'IER=',ier
      end
```

F1

```
      real function f1(x)
c
c   call a unix shell script which executes the BLAKE
c   computer code with input parameter x (% concentration of
c   polyethylene) and returns the value impet (propellant impetus)
c

      real    x, polyet, impet
      character*60 cmd
      character*20 store1
c
c  using internal writes to convert the floating point
c  numbers to a character string
c
      write( store1, '(g15.6)') x
      cmd='run.blake  '//store1
c
c         unix system call
c
      call SYS(cmd)

      open(9,file='blake.out')
      rewind(9)
      read(9,*) polyet,impet

      write(*,'(a,f10.2)')    ' concentration of poly eth.=: ',polyet
      write(*,'(a,f10.1//)')  '     impetus is: ',impet

      f=-impet
      return
      end
```

blake.in

```
PRL,CON,0,PAG,0
TITLE, POLYETHYLENE PLUS 70% H2O2
FOR, POLYET, -14000, C,2,H,4
FOR,OXID70,-1558.28E6,O,43705,H,56295
COM, POLYET, 8.0000 , OXID70, 92
UNI, ENG
GUN, 0.2,10,0
STOP
```

run.blake

```
#    BLAKE Driver:

    POLY=$1  # Oxidant is ( 100% - Polyethylene concentration )
    OXY=` echo "100.0 - $POLY " | bc `

    sed  's/COM, POLYET, 8.0000/COM, POLYET, '$POLY'/p' <blake.in |
    sed  's/OXID70, 92/OXID70, '$OXY'/p' >temp

    BLAKE < temp > output  2>/dev/null


    # impetus is the 2nd from the last line, in column 5
    IMPETUS=`tail -2 output | sed -n 1p | awk '{ printf "%10.3f", $5 }'`

#
#----------------- FINAL OUTPUT --------------------------
#

echo $POLY      >blake.out
echo $IMPETUS  >>blake.out

#--------------------------------------------------------
```

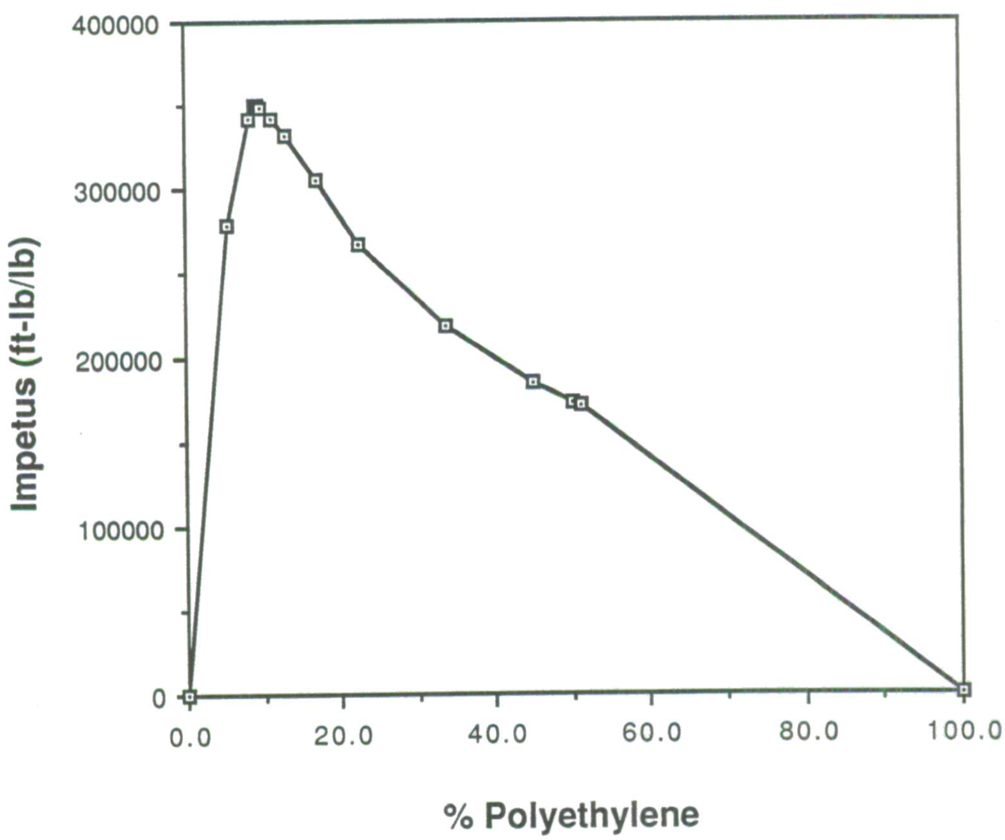

Figure 4. Optimization of Impetus

The second case again adjusts the combination but now we are looking for the lowest concentration of polyethylene which produces no solid carbon as a reaction product (fouls gun mechanisms). The search algorithm is a simple bisection method called directly from the main program.

Table 5. Program Organization for Case 2

Figure 1	*Example 1*	*Description*
MAIN	BRACKET	main program for optimization
SUB1	F2	user-written routine
SYS	sys	C-code to perform system call
Commands	run.blake.2	operating system commands to execute "black box"
MAIN2	BLAKE	"black box" code

BRACKET

```
c  main program
      external f2
      real f2, x0, x1, x, tol, fx

      tol=0.005
      x0=1.0
      x1=99.0

c     repeat until differences are < tol
   10 continue
         x=0.5*(x1+x0)
         fx=f(x)
         if (  fx .lt. tol ) then
            x0=x
         else
            x1=x
         endif
      if ( (x1-x0) .gt. tol ) go to 10
      write (*, '(a,f6.2,a,f15.5)' ) 'X = ', x,'   C(s) =',fx
      end
```

F2

```
      real function f2(x)
c
c   call a unix shell script which execute the blake
c   computer code with input parameter x (% concentration of
c   polyethylene and returns the value CS (solid carbon)
c
      real    x, polyet, cs
      character*60 cmd
      character*20 store1
c
c  using internal writes to convert the floating point
c  numbers to a character string
c
      write( store1, '(g15.6)') x
      cmd='run.blake.2  '//store1
c
c         unix system call
c
      call SYS(cmd)

      open(9,file='blake.out')
      rewind(9)
      read(9,*) polyet,cs
      write(*,'(a,f10.3)')    ' concentration of poly eth.=: ',polyet
      write(*,'(a,f10.3//)')  '             solid carbon is: ',cs
      f=cs
      return
      end
```

run.blake.2

```
#    BLAKE Driver:

POLY=$1  # Oxidant is ( 100% - Polyethylene concentration )
OXY=` echo "100.0 - $POLY " | bc `

sed  's/COM, POLYET, 8.0000/COM, POLYET, '$POLY'/p' <blake2.in |
sed  's/OXID70, 92/OXID70, '$OXY'/p' >temp

BLAKE < temp > output  2>/dev/null

Cs=` grep 'C(S)   SOLID' output | awk '{ printf "%15.5f", $4 }'`

echo $POLY      >blake.out
echo $Cs       >>blake.out
```

blake2.in

```
TITLE, POLYETHYLENE PLUS 70% H2O2
FOR, POLYET, -14000, C,2,H,4
FOR,OXID70,-1558.28E6,O,43705,H,56295
COM, POLYET, 8.0000 , OXID70, 92
UNI, ENG
GUN, 0.2,10,0
STOP
```

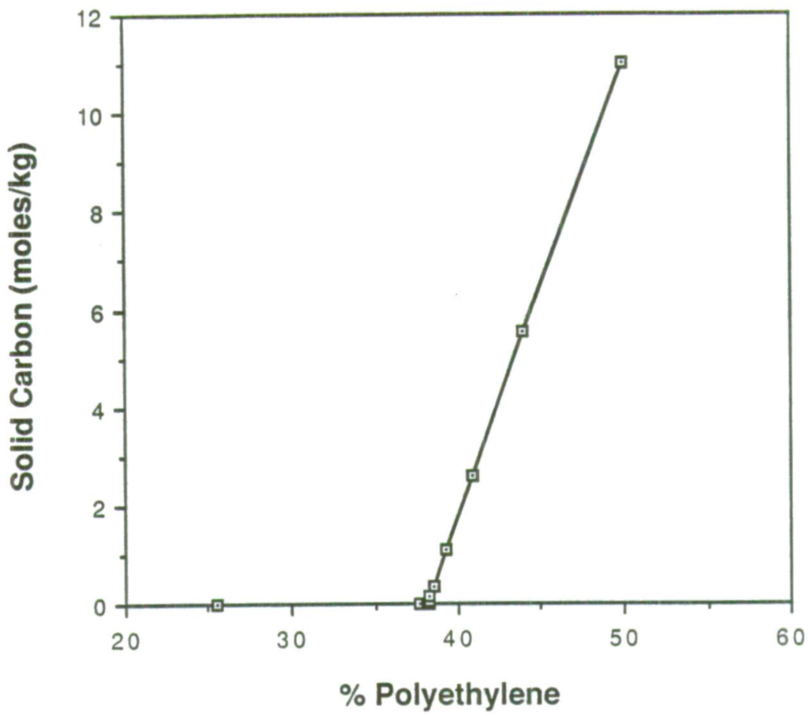

Figure 5. Maximum Concentration without Solid Carbon

III. FURTHER EXTENSIONS

It is hoped that cases described in the previous section are sufficient to illustrate details of the method yet simple enough for the reader to comprehend each problem without undue detail. Practical problems would have main programs of several hundred lines and numerous calls to various subroutine libraries. This section is included to briefly discuss more elaborate applications of this technique which may highlight its strengths and versatility.

A. Parallel Processing

Many computational schemes require a number of similar calculations which are uncoupled from each other. As an illustration, *ab initio* chemistry problems typically require large numbers of individual integrals to be evaluated. Running these independent sub-problems on multiple processing elements in parallel can be quite attractive. The taxonomy of parallel computation is best left to the literature [8]; however, the different approaches are characterized by 1) the number of independent paths to memory with data streams which can be shunted in parallel, and 2) the number of independent instructions which can be executed in parallel. The multiple instruction stream, multiple data stream (MIMD) machine is rapidly becoming the most cost-effective design because the processors tend to operate independently. This permits

an architecture of inexpensive, off-the-shelf computers. The tradeoff is usually a more complex algorithm in order to effectively harness the multiple processors. The problems which work best are those which can be decomposed into distinct processes conditioned to run on separate computers simultaneously. That is, they should require no communication at intermediate steps. All the better if the processes execute in a predictable manner. This alleviates the overhead of synchronizing the results and efficiently loading the multiple processors.

The type of problems discussed in this paper are ideal for this situation. Separate "black box" runs could be spawned simultaneously and processed independently via distinct computers. The optimization example with IBHVG2 could easily exploit this strategy. A Hessian matrix is computed which requires several predetermined IBHVG2 runs to be executed. A serial machine would do one after the other, while a parallel configuration could do all the calculations for the matrix at the same time. This is illustrated in Figure 6. Furthermore, the execution times of each run of IBHVG2 in this circumstance will be nearly identical. Although the complexity of the main program would increase, the burden should not be too oppressive and the payoff could be substantial.

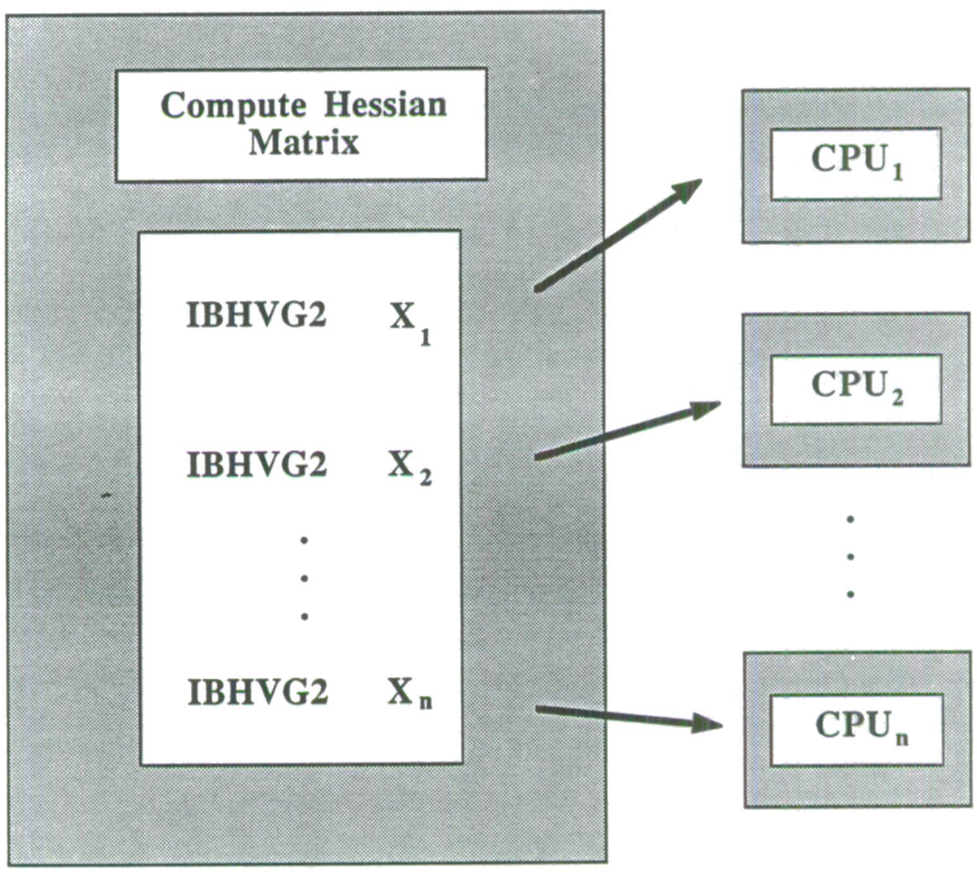

Figure 6. Potential Speedup due to Parallel Processing.

B. Expert Systems

The terms *artificial intelligence* and *expert systems* have become popular buzz words of the 1980's. Most reports and proposals which use them rely more on flash than substance. This is particulary true when the applications wander outside the realm of computer science. The hopes for expert systems are usually the modern day version of a "free lunch"—computers with some magical software are suppose to solve formidable problems without the need of human experts. The question is: *who* writes the software? No successful expert system has avoided the active involvement of the experts. This is why there are so few expert systems which accomplish significant tasks. By definition, *experts* are highly skilled individuals in their chosen disciplines. Convincing them to allocate sizable chunks of time to produce a complex computer program can be both difficult and very expensive. Most commercial expert systems are simply elaborate database programs with little deductive reasoning capability.

This may well change in the next decade or two. A recent resurgence in artificial intelligence has been fostered by the dramatic increase in computing resources and some novel hardware concepts. Neural networks and massively distributed computing are two of the latest topics under scrutiny. Replacing or augmenting experts with software is certainly a long-range, but laudable goal. The commercial impact of synthesizing human reasoning and experience gathering would be enormous. Although researchers clearly have many obstacles to overcome, the potential payoff is sufficient to sustain interest for many years to come.

In the meantime, we suggest a less ambitious approach. Rather than trying to replace the expert, it is probably more fruitful to merely automate routine manipulations. It is possible to utilize the working computational codes of the experts and couple them to another code which could simulate some simple thinking processes. We prefer to call this portion "automated reasoning" to perhaps convey the more limited scope. There are several realizable benefits in a design situation. Reduced labor costs are possible by eliminating most of the user interaction during the design iteration cycles. Enhanced designs are possible because the computer can systematically examine numerous designs far beyond the patience and endurance of a human operator. An intelligent interface can minimize errors by screening information for consistency, satisfy constraints, etc. If nothing else, the computational modules can be made less cumbersome by driving them with a user-friendly environment. A simple mock-up is illustrated in Figure 7.

There is a fundamental difference between this use of artificial intelligence and the applications one sees in magazines or on television. The popularized programs are *knowledge-based* in that they contain a very complex database of knowledge distilled from experts. Elaborate and sophisticated logical inferences require fast manipulation of the database which may be filled with qualitative and subjective information. An example of this type program is an expert system for medical diagnosis. Our approach is compute intensive as opposed to logic intensive. The "intelligence" resides mainly in the experts' computational codes. The computer runs them much like the human would—just thousands of times faster. It is analogous to observing an experiment for quantitative information. In fact, the computer models usually are a distillation of experiments themselves and they simulate certain features which we understand. As opposed to knowledge-based, we refer to this type of expert system as *analysis-based*.

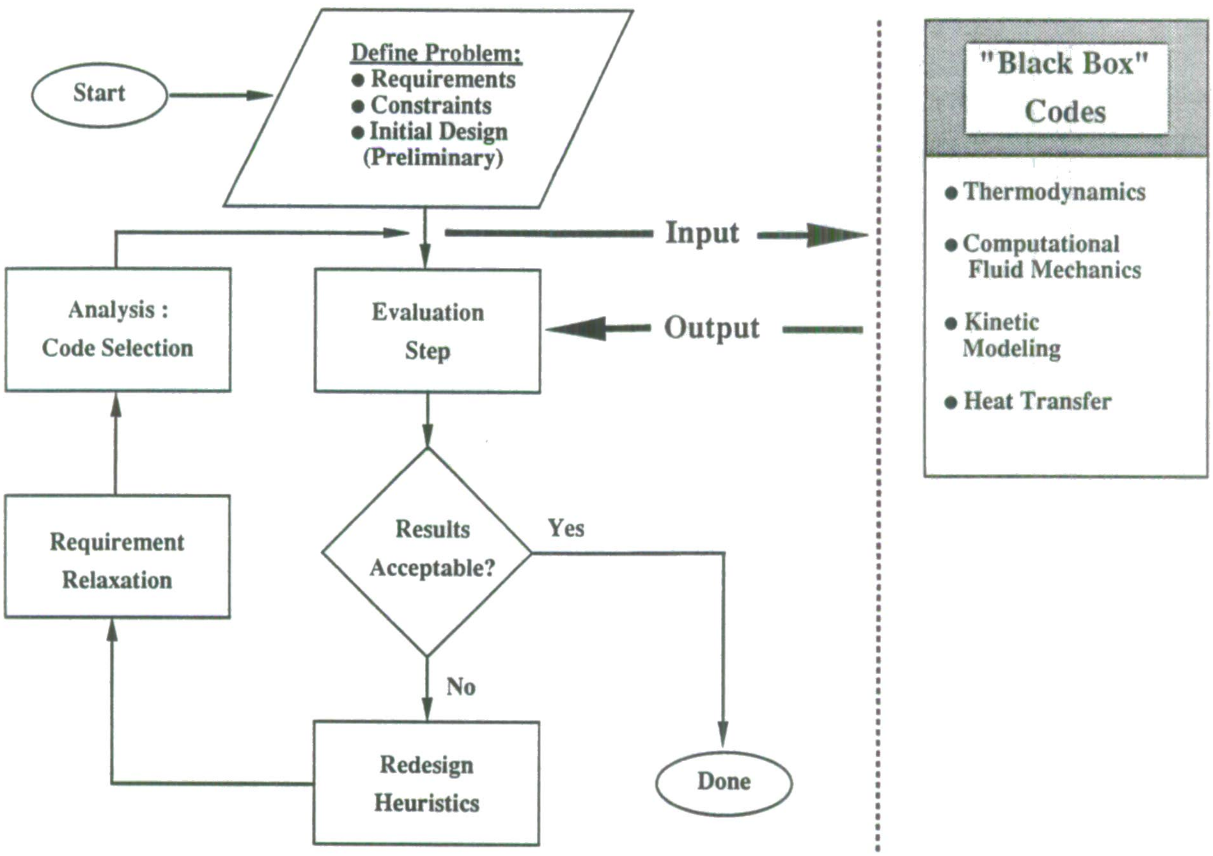

Figure 7. Proposed System for Charge Design

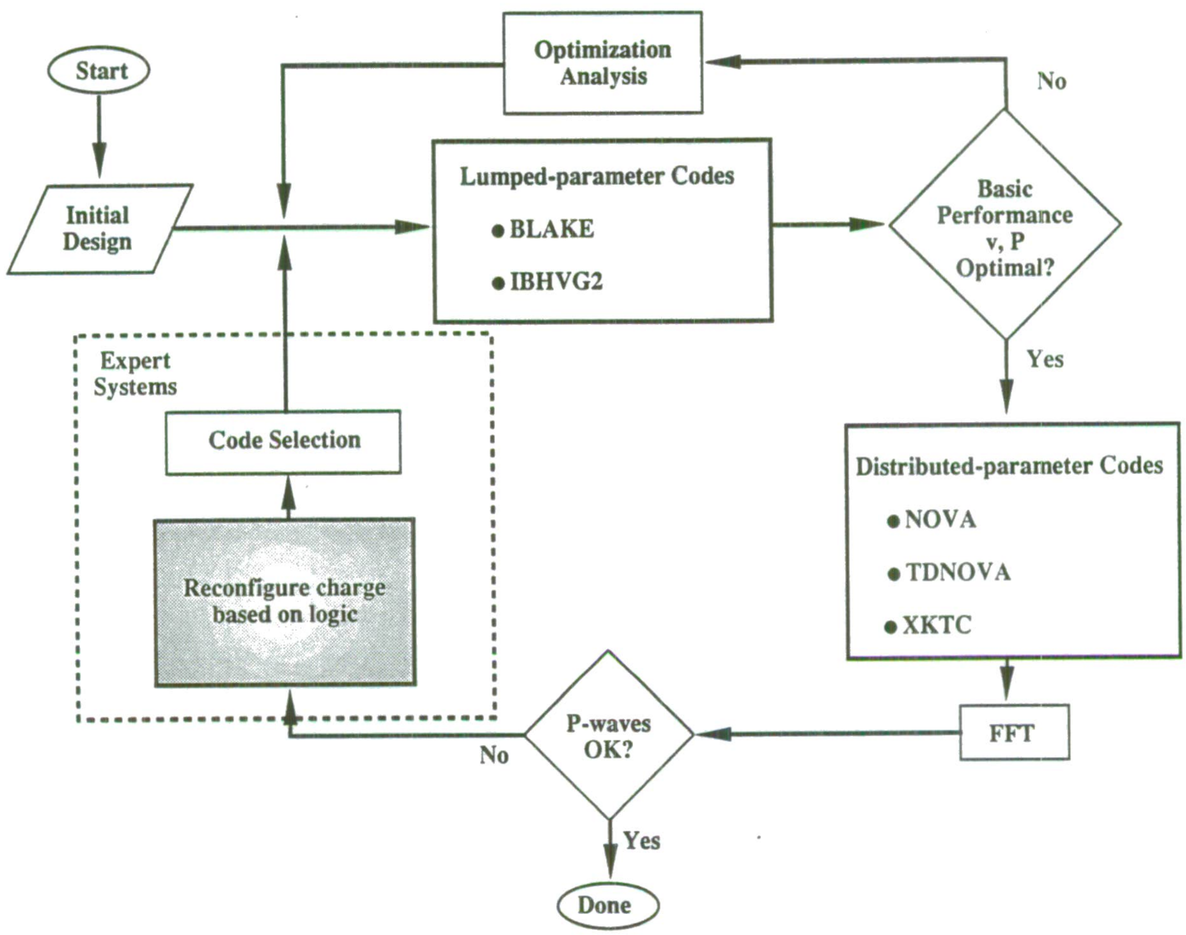

Figure 8. Sample Design Strategy

Logical inferences could be made without introducing the specter of artifical intelligence (AI). The reasoning section could be implemented in a FORTRAN code just by constructing intricate, nested If-Then-Else statements. However, AI languages such as LISP and PROLOG are far more suitable to handle logic. The key features which make these languages ideal is the ability to manipulate program source code as data (self-modifying), the capability to manipulate symbols as well as numbers, and finally, dynamic allocation of storage. Together, these advantages produce concise, straight-forward programs which can be molded to applications easily. Most expert systems are rule-based. This means that the execution and communication of different modules can string together much like theorems in mathematics. Furthermore, there is usually a way of tracing through the series of rules to examine how the program arrived at a particular result. This is called *back-tracking*. It is extremely useful for debugging and verifying the logic. We will not go into the details of AI logic programming here. The main point is that coupling an AI language with FORTRAN analysis codes can be effectively implemented using the techniques described in the report.

Let us conjure up a possible ballistics example. Suppose one needed to design a propulsion system for a nuclear round. Besides the standard performance requirements of zone charges and muzzle velocities with range overlap, great care is to be paid to pressure waves due to the nature of the payload. The design strategy might look something like Figure 8.

An initial charge configuration is provided as input to both BLAKE and IBHVG2 to compute and optimize the gross ballistics of the round (mean pressure and velocity history). These codes would not be capable of predicting pressure waves, however. At an appropriate point, a distributed-parameter code such as NOVA would be needed. The resulting wave behavior could be evaluated by computing the Fourier transform of the breech pressure history. A power spectrum beyond a specified norm could flag an unacceptable design. Depending on the results, the charge could be modified by a rule-based expert system. For example, pressure waves might be ameliorated by increasing the ullage or by altering the ignition system. Radical changes such as propellant reformulation would require going all the way back to the lumped-parameter codes, while perturbations could iterate using the distributed codes.

IV. CONCLUSIONS

A simple method of tying together existing computer programs has been both demonstrated and explained. The implementation involves system calls to the operating system. Although the examples in this paper have focussed on the UNIX operating system, the technique is general and should be applicable to many other operating systems. Some advantages to this approach include increased modularity, added productivity, and the potential for more elaborate applications. It is hoped the three examples provided in the text have demonstrated the first two virtues. The last advantage was only discussed and is still to be proven. A convincing application will be the subject of a future technical report.

REFERENCES

[1] R. Anderson and K. Fickie, "IBHVG2—A User's Guide," BRL-TR-2829, Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1987.

[2] E. Freedman, "BLAKE—A Thermodynamics Code Based on TIGER: User's Guide and Manual," ARBRL-TR-02411, Ballistic Research Laboratory, Aberdeen Proving Ground, MD, 1982.

[3] **IMSL Library Reference Manual,** Edition 9.2, (IMSL Inc., 7500 Bellaire Boulevard, Houston TX 77036) Volume 4, Chapter Z, 1984.

[4] G. Strang, **Introduction to Applied Mathematics,** Wellesley-Cambridge Press (Wellesley, 1986).

[5] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, **Numerical Recipes,** Cambridge University Press (New York, 1986).

[6] P.E. Gill, W. Murray, M. H. Wright, **Practical Optimization,** Academic Press (New York, 1981).

[7] E. Polak, **Computational Methods in Optimization,** Academic Press (New York, 1971).

[8] G. Rodrigue, **Parallel Computations,** Academic Press (New York, 1982).

APPENDIX

As briefly mentioned in the report, the UNIX operating system provides additional ways of executing system commands inside an active process (i.e., a separate executing computer program). One technique would be to use *pipes*. Pipes allow transfer of data streams between processes and can be used for process synchronization. Their most useful characteristic for inter-process communication is: streams need no information about what processes are at the other end of the pipeline. There are two types of pipes. The simplest is a *system pipe* shown in Figure 9 (some people call these *unnamed pipes*). Basically, they work much like files, but the UNIX kernel uses direct blocks of the inode for greater efficiency. The other type is a *process pipe* (sometimes called a *named pipe*). Figure 10 shows typical code fragments using process pipes.

It would be inappropriate to fully discuss the mechanisms and added virtues of using pipes for the applications described in this paper. The discussion would be rather involved and to fully comprehend the arguments requires some background in UNIX system programming. We mention them because a subroutine call to a "black box" process using pipes could more closely emulate a normal subroutine call written in a standard language such as FORTRAN. Pipes can both send values and return values transparently without resorting to external files. They would provide an elegant interface between processes, and potentially more sophisticated and capable. We relegate this idea to an appendix because the method is very specific to UNIX. See your system programmer for more details.

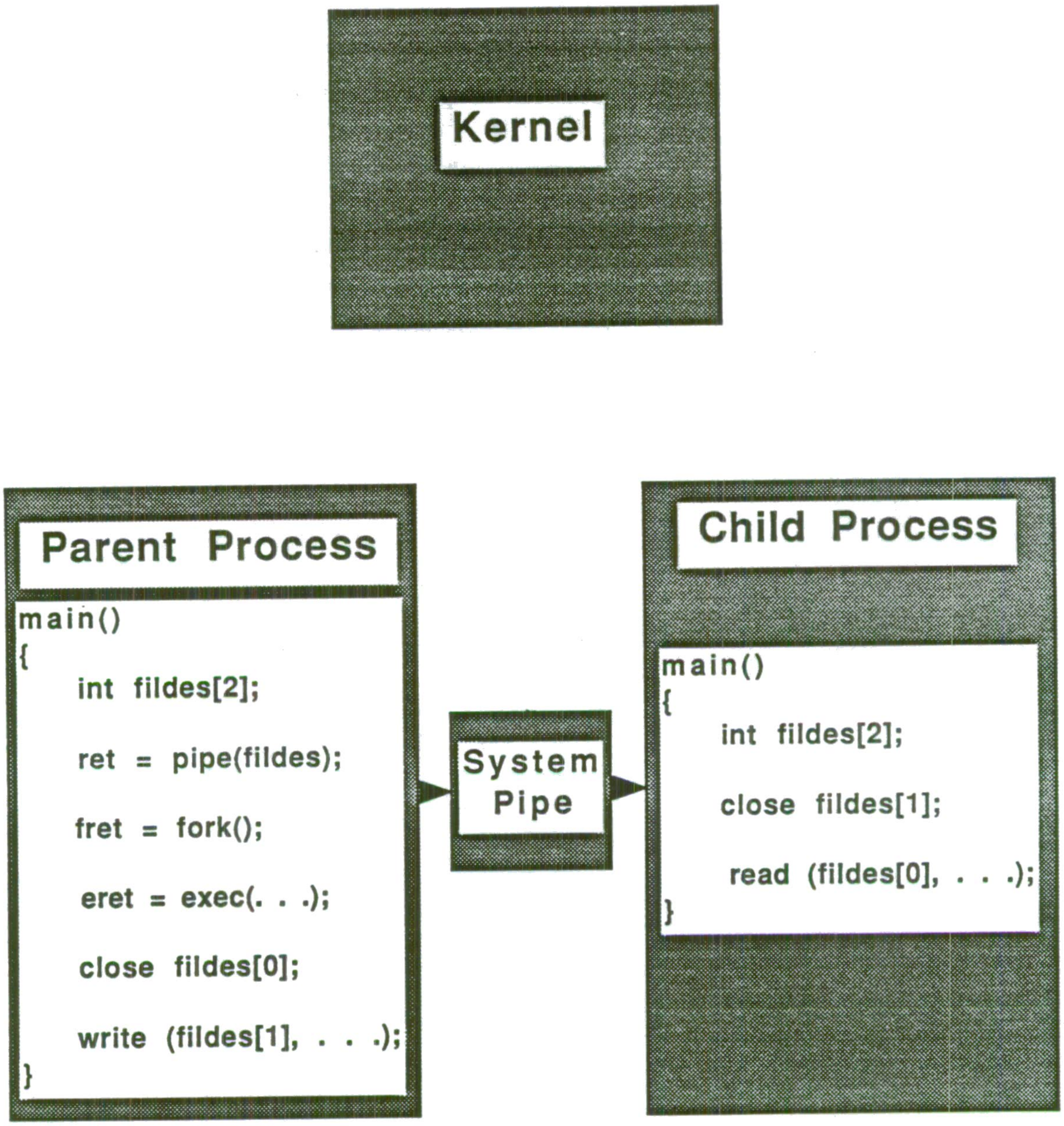

Figure 9. A System Pipe

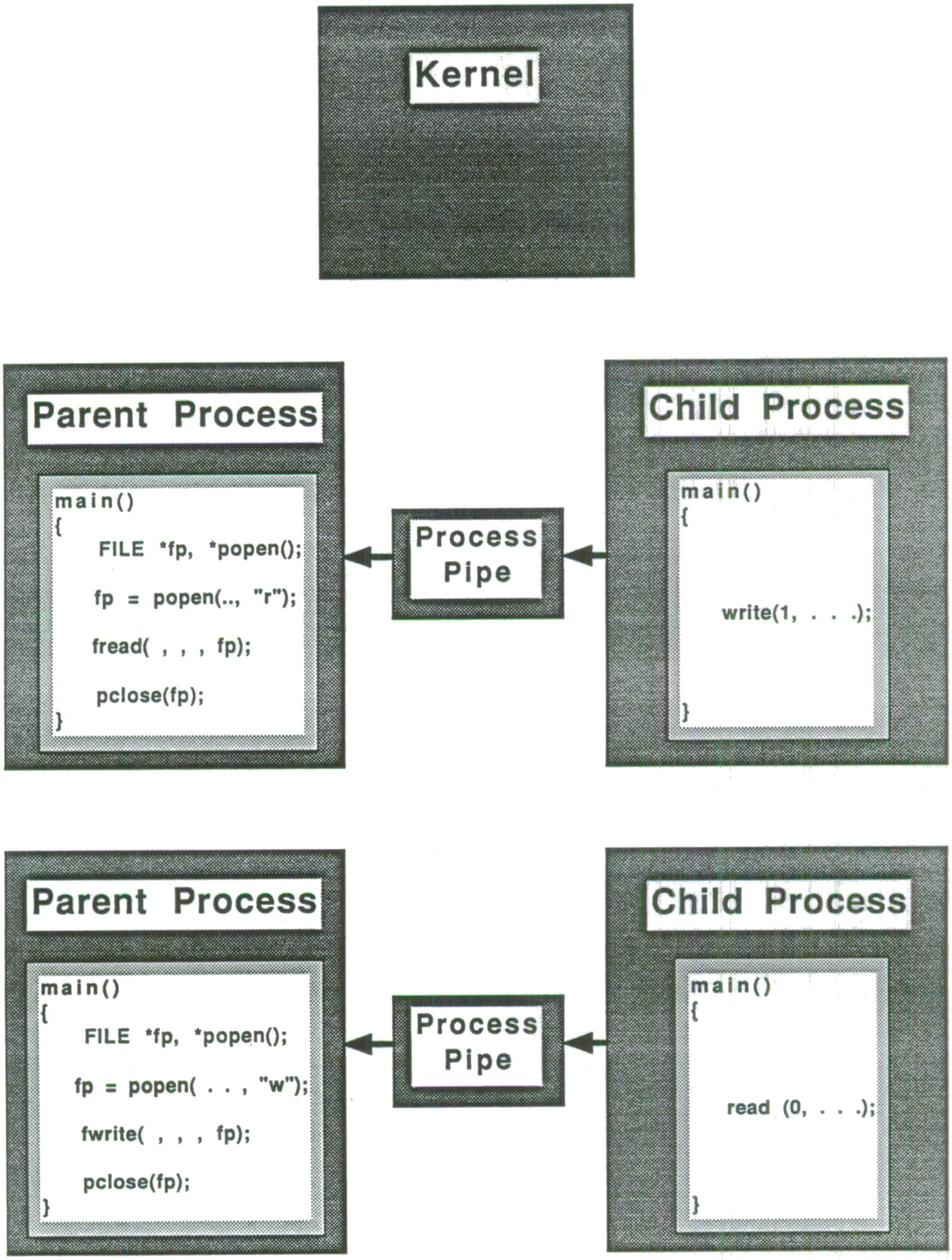

Figure 10. Process Pipes

DISTRIBUTION LIST

No. of Copies	Organization
12	Administrator Defense Technical Info Center ATTN: DTIC-FDAC Cameron Station, Bldg 5 Alexandria, VA 22304-6145
1	Commander USA Concepts Analysis Agency ATTN: D. Hardison 8120 Woodmont Avenue Bethesda, MD 20014-2797
1	HQDA/DAMA-ZA Washington, DC 20310-2500
1	HQDA, DAMA-CSM, Washington, DC 20310-2500
1	HQDA/SARDA Washington, DC 20310-2500
1	C.I.A. OIR/DB/Standard GE47 HQ Washington, D.C. 20505
1	Commander US Army War College ATTN: Library-FF229 Carlisle Barracks, PA 17013
1	US Army Ballistic Missile Defense Systems Command Advanced Technology Center P. O. Box 1500 Huntsville, AL 35807-3801
1	Chairman DOD Explosives Safety Board Room 856-C Hoffman Bldg. 1 2461 Eisenhower Avenue Alexandria, VA 22331-9999
1	Commander US Army Materiel Command ATTN: AMCPM-GCM-WF 5001 Eisenhower Avenue Alexandria, VA 22333-5001
1	Commander US Army Materiel Command ATTN: AMCDRA-ST 5001 Eisenhower Avenue Alexandria, VA 22333-5001
1	Commander US Army Materiel Command ATTN: AMCDE-DW 5001 Eisenhower Avenue Alexandria, VA 22333-5001
5	Project Manager Cannon Artillery Weapons System, ARDC, AMCCOM ATTN: AMCPM-CW, AMCPM-CWW AMCPM-CWS M. Fisette AMCPM-CWA H. Hassmann AMCPM-CWA-S R. DeKleine Dover, NJ 07801-5001
2	Project Manager Munitions Production Base Modernization and Expansion ATTN: AMCPM-PBM, A. Siklosi AMCPM-PBM-E, L. Laibson Dover, NJ 07801-5001
3	Project Manager Tank Main Armament System ATTN: AMCPM-TMA, K. Russell AMCPM-TMA-105 AMCPM-TMA-120 Dover, NJ 07801-5001
1	Commander US Army Watervliet Arsenal ATTN: SARWV-RD, R. Thierry Watervliet, NY 12189-5001
1	Commander U.S. Army ARDEC ATTN: SMCAR-MSI Dover, NJ 07801-5001
1	Commander U.S. Army ARDEC ATTN: SMCAR-TDC Dover, NJ 07801-5001
4	Commander US Army Armament Munitions and Chemical Command ATTN: AMSMC-IMP-L Rock Island, IL 61299-7300
1	HQDA DAMA-ART-M Washington, DC 20310-2500
1	Commander US Army AMCCOM ARDEC CCAC ATTN: SMCAR-CCB-TL Benet Weapons Laboratory Watervliet, NY 12189-4050

DISTRIBUTION LIST

No. of Copies	Organization
3	Commander US Army ARDEC ATTN: SMCAR-MSI SMCAR-TDC SMCAR-LC LTC N. Barron Dover, NJ 07801-5001
7	Commander US Army ARDEC ATTN: SMCAR-LCA A. Beardell D. Downs S. Einstein S. Westley S. Bernstein C. Roller J. Rutkowski Dover, NJ 07801-5001
3	Commander US Army ARDEC ATTN: SMCAR-LCB-I D. Spring SMCAR-LCE SMCAR-LCM-E S. Kaplowitz Dover, NJ 07801-5001
4	Commander US Army ARDEC ATTN: SMCAR-LCS SMCAR-LCU-CT E. Barrieres R. Davitt SMCAR-LCU-CV C. Mandala Dover, NJ 07801-5001
3	Commander US Army ARDEC ATTN: SMCAR-LCW-A M. Salsbury SMCAR-SCA L. Stiefel B. Brodman Dover, NJ 07801-5001
1	Commander US Army Aviation Systems Command ATTN: AMSAV-ES 4300 Goodfellow Blvd. St. Louis, MO 63120-1798
1	Director US Army Aviation Research and Technology Activity Ames Research Center Moffett Field, CA 94035-1099

No. of Copies	Organization
1	Commander US Army Communications - Electronics Command ATTN: AMSEL-ED Fort Monmouth, NJ 07703-5301
1	Commander CECOM R&D Technical Library ATTN: AMSEL-M-L (Report Section) B.2700 Fort Monmouth, NJ 07703-5000
1	Commander US Army Harry Diamond Lab. ATTN: DELHD-TA-L 2800 Powder Mill Road Adelphi, MD 20783-1145
1	Commander US Army Missile Command ATTN: AMSMI-RX M.W. Thauer Redstone Arsenal, AL 35898-5249
1	Commander US Army Missile and Space Intelligence Center ATTN: AIAMS-YDL Redstone Arsenal, AL 35898-5500
1	Commander US Army Missile Command Research, Development, and Engineering Center ATTN: AMSMI-RD Redstone Arsenal, AL 35898-5245
1	Commandant US Army Aviation School ATTN: Aviation Agency Fort Rucker, AL 36360
1	Commander US Army Tank Automotive Command ATTN: AMSTA-TSL Warren, MI 48397-5000
1	Commander US Army Tank Automotive Command ATTN: AMSTA-CG Warren, MI 48397-5000

DISTRIBUTION LIST

No. of Copies	Organization
1	Project Manager Improved TOW Vehicle ATTN: AMCPM-ITV US Army Tank Automotive Command Warren, MI 48397-5000
2	Program Manager M1 Abrams Tank System ATTN: AMCPM-GMC-SA, T. Dean Warren, MI 48092-2498
1	Project Manager Fighting Vehicle Systems ATTN: AMCPM-FVS Warren, MI 48092-2498
1	President US Army Armor & Engineer Board ATTN: ATZK-AD-S Fort Knox, KY 40121-5200
1	Project Manager M-60 Tank Development ATTN: AMCPM-M60TD Warren, MI 48092-2498
1	Director US Army TRADOC Systems Analysis Activity ATTN: ATOR-TSL White Sands Missile Range, NM 88002
1	Commander US Army Training & Doctrine Command ATTN: ATCD-MA/ MAJ Williams Fort Monroe, VA 23651
2	Commander US Army Materials and Mechanics Research Center ATTN: AMXMR-ATL Tech Library Watertown, MA 02172
1	Commander US Army Research Office ATTN: Tech Library P. O. Box 12211 Research Triangle Park, NC 27709-2211
1	Commander US Army Belvoir Research and Development Center ATTN: STRBE-WC Fort Belvoir, VA 22060-5606

No. of Copies	Organization
1	Commander US Army Logistics Mgmt Ctr Defense Logistics Studies Fort Lee, VA 23801
1	Commandant US Army Infantry School ATTN: ATSH-CD-CS-OR Fort Benning, GA 31905-5400
1	Commandant US Army Command and General Staff College Fort Leavenworth, KS 66027
1	Commandant US Army Special Warfare School ATTN: Rev & Tng Lit Div Fort Bragg, NC 28307
3	Commander Radford Army Ammunition Plant ATTN: SMCRA-QA/HI LIB Radford, VA 24141-0298
1	Commander US Army Foreign Science & Technology Center ATTN: AMXST-MC-3 220 Seventh Street, NE Charlottesville, VA 22901-5396
2	Commandant US Army Field Artillery Center & School ATTN: ATSF-CO-MW, B. Willis Ft. Sill, OK 73503-5600
1	Commander US Army Development and Employment Agency ATTN: MODE-ORO Fort Lewis, WA 98433-5099
1	Office of Naval Research ATTN: Code 473, R. S. Miller 800 N. Quincy Street Arlington, VA 22217-9999
3	Commandant US Army Armor School ATTN: ATZK-CD-MS M. Falkovitch Armor Agency Fort Knox, KY 40121-5215

DISTRIBUTION LIST

No. of Copies	Organization
2	Commander Naval Sea Systems Command ATTN: SEA 62R SEA 64 Washington, DC 20362-5101
1	Commander Naval Air Systems Command ATTN: AIR-954-Tech Lib Washington, DC 20360
1	Assistant Secretary of the Navy (R, E, and S) ATTN: R. Reichenbach Room 5E787 Pentagon Bldg. Washington, DC 20350
1	Naval Research Lab Tech Library Washington, DC 20375
5	Commander Naval Surface Weapons Center ATTN: Code G33, J. L. East W. Burrell J. Johndrow Code G23, D. McClure Code DX-21 Tech Lib Dahlgren, VA 22448-5000
2	Comander US Naval Surface Weapons Center ATTN: J. P. Consaga C. Gotzmer Indian Head, MD 20640-5000
4	Commander Naval Surface Weapons Center ATTN: S. Jacobs/Code 240 Code 730 K. Kim/Code R-13 R. Bernecker Silver Spring, MD 20903-5000
2	Commanding Officer Naval Underwater Systems Center Energy Conversion Dept. ATTN: CODE 5B331, R. S. Lazar Tech Lib Newport, RI 02840
4	Commander Naval Weapons Center ATTN: Code 388, R. L. Derr C. F. Price T. Boggs Info. Sci. Div. China Lake, CA 93555-6001

No. of Copies	Organization
2	Superintendent Naval Postgraduate School Dept. of Mech. Engineering Monterey, CA 93943-5100
1	Program Manager AFOSR Directorate of Aerospace Sciences ATTN: L. H. Caveny Bolling AFB, DC 20332-0001
5	Commander Naval Ordnance Station ATTN: P. L. Stang L Torreyson T. C. Smith D. Brooks Tech Library Indian Head, MD 20640-5000
1	AFSC/SDOA Andrews AFB, MD 20334
3	AFRPL/DY, Stop 24 ATTN: J. Levine/DYCR R. Corley/DYC D. Williams/DYCC Edwards AFB, CA 93523-5000
1	AFRPL/TSTL (Tech Library) Stop 24 Edwards AFB, CA 93523-5000
1	AFATL/DLYV Eglin AFB, FL 32542-5000
1	AFATL/DLXP Eglin AFB, FL 32542-5000
1	AFATL/DLJE Eglin AFB, FL 32542-5000
1	AFATL/DOIL ATTN: (Tech Info Center) Eglin AFB, FL 32542-5438
1	NASA/Lyndon B. Johnson Space Center ATTN: NHS-22, Library Section Houston, TX 77054
1	AFELM, The Rand Corporation ATTN: Library D 1700 Main Street Santa Monica CA 90401-3297

DISTRIBUTION LIST

No. of Copies	Organization
1	General Applied Sciences Lab ATTN: J. Erdos Merrick & Stewart Avenues Westbury Long Isld, NY 11590
2	AAI Corporation ATTN: J. Hebert J. Frankle D. Cleveland P. O. Box 6767 Baltimore, MD 21204
1	Aerojet Ordnance Company ATTN: D. Thatcher 2521 Michelle Drive Tustin, CA 92680-7014
1	Aerojet Solid Propulsion Co. ATTN: P. Micheli Sacramento, CA 95813
1	Atlantic Research Corporation ATTN: M. K. King 5390 Cheorokee Avenue Alexandria, VA 22312-2302
1	AVCO Everett Rsch Lab ATTN: D. Stickler 2385 Revere Beach Parkway Everett, MA 02149-5936
2	Calspan Corporation ATTN: C. Morphy P. O. Box 400 Buffalo, NY 14225-0400
1	General Electric Company Armament Systems Dept. ATTN: M. J. Bulman, Room 1311 128 Lakeside Avenue Burlington, VT 05401-4985
1	IITRI ATTN: M. J. Klein 10 W. 35th Street Chicago, IL 60616-3799
1	Hercules Inc. Allegheny Ballistics Laboratory ATTN: R. B. Miller P. O. Box 210 Cumberland, MD 21501-0210
1	Hercules, Inc. Bacchus Works ATTN: K. P. McCarty P. O. Box 98 Magna, UT 84044-0098
1	Hercules, Inc. Radford Army Ammunition Plant ATTN: J. Pierce Radford, VA 24141-0299
1	Honeywell, Inc. - MN64 2200 Defense Systems Division ATTN: C. Hargreaves 6110 Blue Circle Drive Minnetonka MN 55436
1	Lawrence Livermore National Laboratory ATTN: L-355, A. Buckingham M. Finger P. O. Box 808 Livermore, CA 94550-0622
1	Lawrence Livermore National Laboratory ATTN: L-324/M. Constantino P. O. Box 808 Livermore, CA 94550-0622
1	Olin Corporation Badger Army Ammunition Plant ATTN: R. J. Thiede Baraboo, WI 53913
1	Olin Corporation Smokeless Powder Operations ATTN: D. C. Mann P.O. Box 222 St. Marks, FL 32355-0222
1	Paul Gough Associates, Inc. ATTN: P. S. Gough P. O. Box 1614, 1048 South St. Portsmouth, NH 03801-1614
1	Physics International Company ATTN: Library H. Wayne Wampler 2700 Merced Street San Leandro, CA 94577-5602
1	Princeton Combustion Research Lab., Inc. ATTN: M. Summerfield 475 US Highway One Monmouth Junction, NJ 08852-9650
2	Rockwell International Rocketdyne Division ATTN: BA08 J. E. Flanagan J. Gray 6633 Canoga Avenue Canoga Park, CA 91303-2703

DISTRIBUTION LIST

No. of Copies	Organization
1	Science Applications, Inc. ATTN: R. B. Edelman 23146 Cumorah Crest Drive Woodland Hills, CA 91364-3710
3	Thiokol Corporation Huntsville Division ATTN: D. Flanigan R. Glick Tech Library Huntsville, AL 35807
2	Thiokol Corporation Elkton Division ATTN: R. Biddle Tech Lib. P. O. Box 241 Elkton, MD 21921-0241
2	United Technologies Chemical Systems Division ATTN: R. Brown Tech Library P. O. Box 358 Sunnyvale, CA 94086-9998
1	Veritay Technology, Inc. ATTN: E. Fisher 4845 Millersport Hwy. P. O. Box 305 East Amherst, NY 14051-0305
1	Universal Propulsion Company ATTN: H. J. McSpadden Black Canyon Stage 1 Box 1140 Phoenix, AZ 85029
1	Battelle Memorial Institute ATTN: Tech Library 505 King Avenue Columbus, OH 43201-2693
1	Brigham Young University Dept. of Chemical Engineering ATTN: M. Beckstead Provo, UT 84601
1	California Institute of Tech 204 Karman Lab Main Stop 301-46 ATTN: F. E. C. Culick 1201 E. California Street Pasadena, CA 91109
1	California Institute of Tech Jet Propulsion Laboratory ATTN: L. D. Strand 4800 Oak Grove Drive Pasadena, CA 91109-8099

No. of Copies	Organization
1	University of Illinois Dept of Mech/Indust Engr ATTN: H. Krier 144 MEB; 1206 N. Green St. Urbana, IL 61801-2978
1	University of Massachusetts Dept. of Mech. Engineering ATTN: K. Jakus Amherst, MA 01002-0014
1	University of Minnesota Dept. of Mech. Engineering ATTN: E. Fletcher Minneapolis, MN 55414-3368
1	Case Western Reserve University Division of Aerospace Sciences ATTN: J. Tien Cleveland, OH 44135
3	Georgia Institute of Tech School of Aerospace Eng. ATTN: B. T. Zinn E. Price W. C. Strahle Atlanta, GA 30332
1	Institute of Gas Technology ATTN: D. Gidaspow 3424 S. State Street Chicago, IL 60616-3896
1	Johns Hopkins University Applied Physics Laboratory Chemical Propulsion Information Agency ATTN: T. Christian Johns Hopkins Road Laurel, MD 20707-0690
1	Massachusetts Institute of Technology Dept of Mechanical Engineering ATTN: T. Toong 77 Massachetts Avenue Cambridge, MA 02139-4307
1	G. M. Faeth Pennsylvania State University Applied Research Laboratory University Park, PA 16802-7501
1	Pennsylvania State University Dept. of Mech. Engineering ATTN: K. Kuo University Park, PA 16802-7501

DISTRIBUTION LIST

No. of Copies	Organization
1	Purdue University School of Mechanical Engineering ATTN: J. R. Osborn TSPC Chaffee Hall West Lafayette, IN 47907-1199
1	SRI International Propulsion Sciences Division ATTN: Tech Library 333 Ravenswood Avenue Menlo Park, CA 94025-3493
1	Rensselaer Polytechnic Inst. Department of Mathematics Troy, NY 12181
2	Director Los Alamos Scientific Lab ATTN: T3, D. Butler M. Division, B. Craig P. O. Box 1663 Los Alamos, NM 87544
1	Stevens Institute of Technology Davidson Laboratory ATTN: R. McAlevy, III Castle Point Station Hoboken, NJ 07030-5907
1	Rutgers University Dept. of Mechanical and Aerospace Engineering ATTN: S. Temkin University Heights Campus New Brunswick, NJ 08903
1	University of Southern California Mechanical Engineering Dept. ATTN: 0HE200, M. Gerstein Los Angeles, CA 90089-5199
2	University of Utah Dept. of Chemical Engineering ATTN: A. Baer G. Flandro Salt Lake City, UT 84112-1194
1	Washington State University Dept. of Mech. Engineering ATTN: C. T. Crowe Pullman, WA 99163-5201

Aberdeen Proving Ground

Dir, USAMSAA
ATTN: AMXSY-D
AMXSY-MP, H. Cohen

No. of Copies	Organization
	Cdr, USATECOM ATTN: AMSTE-SI-F AMSTE-CM-F, L. Nealley
	Cdr, CSTA ATTN: STECS-AS-H, R. Hendricksen
	Cdr, CRDC, AMCCOM ATTN: SMCCR-RSP-A SMCCR-MU SMCCR-SPS-IL

USER EVALUATION SHEET/CHANGE OF ADDRESS

This Laboratory undertakes a continuing effort to improve the quality of the reports it publishes. Your comments/answers to the items/questions below will aid us in our efforts.

1. BRL Report Number____________________Date of Report__________

2. Date Report Received______________________________

3. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which the report will be used.)__________

4. How specifically, is the report being used? (Information source, design data, procedure, source of ideas, etc.)__________

5. Has the information in this report led to any quantitative savings as far as man-hours or dollars saved, operating costs avoided or efficiencies achieved, etc? If so, please elaborate.__________

6. General Comments. What do you think should be changed to improve future reports? (Indicate changes to organization, technical content, format, etc.)

CURRENT ADDRESS

Name

Organization

Address

City, State, Zip

7. If indicating a Change of Address or Address Correction, please provide the New or Correct Address in Block 6 above and the Old or Incorrect address below.

OLD ADDRESS

Name

Organization

Address

City, State, Zip

(Remove this sheet, fold as indicated, staple or tape closed, and mail.)

FOLD HERE

Director
US Army Ballistic Research Laboratory
ATTN: DRXBR-OD-ST
Aberdeen Proving Ground, MD 21005-5066

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO 12062 WASHINGTON, DC

POSTAGE WILL BE PAID BY DEPARTMENT OF THE ARMY

Director
US Army Ballistic Research Laboratory
ATTN: DRXBR-OD-ST
Aberdeen Proving Ground, MD 21005-9989

FOLD HERE